# PETSc: Next Generation Readiness Background

- PETSc provides
  - Algebraic solvers
  - Integrators
  - Adjoints
  - Optimization (Tao)
- High level libraries (libMesh, Moose, Firedrake) and application codes provide the rest
- PETSc is organized as a class library. “All” numerical computation and memory usage occurs within
  - Vector classes
  - Matrix classes
  - code under the users responsibility

# PETSc: Next Generation Readiness Computer Model

- MPI based
- Fat nodes
  - N shared memory CPUs
  - m < N physical accelerators with significant memory
  - m <= M < N virtual accelerators
  - Accelerators provide 95+% of the system's
    - Computation performance (flops)
    - Memory bandwidth
  - Bandwidth between CPUs and accelerators low
  - MPI may or may not connect directly to accelerator memory (optimization)
  - Local accelerators may or may not connect directly to neighboring accelerators (optimization)

# PETSc: Next Generation Readiness
# PETSc Programming Model (1)
# Application Codes

- PETSc
- OpenMP 5
- Other higher node level programming models??
- Kokkos
- Lower level programming models
  - CUDA
  - OpenCL
  - Other lower level programming models ??

# PETSc: Next Generation Readiness
# PETSc Programming Model (2)
# PETSc

- Orchestration code (C) runs on the M (MPI rank) CPUs
- Vector and matrix class runs on the
  - N CPUs (OpenMP) (limited usage)
  - M virtualized accelerators
    - CUDA
    - OpenCL
    - ???
- Vector and matrix classes transparently manage data motion (or perceived) motion between CPU and accelerator memory via
  - VecGetArray(), VecGetArrayRead(), VecGetArrayWrite()
  - VecCUDAGetArray(), VecCUDAGetArrayRead(), VecCUDAGetArrayWrite()

# PETSc: Next Generation Readiness
# PETSc Programming Model (3)

- Goal:
  - Essentially “all” PETSc and “all” user computation and data reside on the accelerators.
  - CPUs orchestrate the computation and do the little work that cannot be done on the accelerators
- Route:
  - Move more PETSc matrix operations to accelerators
  - Optimize accelerator code
  - Additional implementations of accelerator code based on ??
  - Fuse vector and matrix methods in accelerator code for higher performance
  - Optimize MPI by pass for local accelerators and send directly from accelerator memory